Didattica del ragionamento: un esempio di deduzione logica in ambiente *Mathematica*.

(Cristina Coppola, Giangiacomo Gerla, Tiziana Pacelli)
Dipartimento di Matematica e Informatica,
Università degli Studi di Salerno,
via Ponte don Melillo, Fisciano (*SA*)*, Italia*
Email: {ccoppola | gerla | tpacelli}@unisa.it

Riassunto. Si mostra come, in un ambiente adeguato per il calcolo simbolico, si possano introdurre elementi di programmazione logica, "traducendo" una teoria in un insieme di definizioni di funzioni. Dal punto di vista didattico, tale riduzione della programmazione logica a programmazione funzionale risulta utile per l'ampliamento dell'uso di strumenti informatici all'ambito della logica matematica. Inoltre, un ulteriore obiettivo è quello di facilitare, attraverso tali strumenti, lo sviluppo di competenze non solo nell'ambito del calcolo numerico-algebrico ma anche in quello dell' informazione di tipo assertivo-logico. Il nostro punto di vista è che nella sperimentazione didattica sia possibile riferirsi alla logica non solo per quanto riguarda il calcolo proposizionale ma anche al suo aspetto deduttivo. E' necessario evidenziare come il metodo assiomatico sia presente nella vita di tutti i giorni. Nozioni come quelle di "sistema di assiomi" e di "teorema" devono essere interpretate in termini di informazione disponibile, (in generale non completa) circa situazioni della vita reale, e di conseguenze che si possono ricavare a partire da tale informazione.

Abstract. We sketch how we can introduce notions of logic programming in an environment commonly used for symbolic calculus. We do this by "rewriting" a theory in order to obtain a set of definitions of functions. From an educational point of view, turning logic programming into functional programming is useful for the extension of information tools in a logical framework. Moreover, by these information systems, we intend to promote the development of logical competences, not only algebraic and numerical ones. We suppose that in teaching methodology we

can introduce logic by referring to its deductive method. Deductive reasoning and axiomatic method are typical of real-life situations. Basic notions, such as “system of axiom” and “theorem” have to be interpreted as pieces of the available (maybe incomplete) information about a real-life situation, and as consequences deduced from this information, respectively.

1 Introduzione

La logica matematica si introduce nell’insegnamento scolastico principalmente in relazione alle tavole di verità, ai circuiti logici, alla logica proposizionale. Nonostante l’utilità di trattare tali argomenti sia pienamente condivisibile, ci si pone, tuttavia, il problema di capire se le potenzialità didattiche della logica matematica non si possano cogliere anche in altri suoi aspetti. In particolare, vogliamo riferirci alla dimensione deduttiva della logica del primo ordine ed al metodo assiomatico, che trova in tale logica l’ambiente naturale in cui svilupparsi. Una risposta a tale problema comporta l’esigenza di individuare all’interno della logica del primo ordine, che spesso si presenta troppo complessa ed astratta, “pezzi” che siano abbastanza semplici da prestarsi alla sperimentazione didattica.

A tale riguardo pensiamo che la prima scelta da fare sia quella di svincolare la logica dalla tradizione fondazionale che, da Hilbert in poi, ha visto in essa lo strumento per la costruzione di una base sicura per l’intero edificio matematico. Non ci sembra nemmeno opportuno riferirsi alla tradizione strutturalista, secondo cui un sistema di assiomi ha il ruolo di individuare e trattare in modo unificato una classe di strutture. In altre parole, non intendiamo vedere la logica ed il metodo assiomatico come strumenti per fondare la geometria o la teoria degli insiemi e nemmeno per elaborare la teoria dei gruppi, degli anelli od altro. Il nostro punto di vista è, invece, che nella sperimentazione didattica, nozioni come quelle di “sistema

di assiomi" e di "teorema" debbano essere interpretate rispettivamente in termini di:

- informazione disponibile (in generale non completa) circa situazioni della vita reale;
- conseguenze che si possono ricavare da tale informazione.

In questo modo, infatti, si evidenzia che il metodo assiomatico è presente nella vita di tutti i giorni e non è distinguibile dalla semplice attività di ragionare. In definitiva, il tipo di attività didattica che si vuole proporre consiste nel

- fissare un semplice sistema di assiomi che rappresenti l'informazione disponibile circa una situazione reale;
- porre agli studenti il problema di valutare se un dato fatto può essere dedotto o no da tale sistema;
- individuare, insieme agli studenti, possibili strategie dimostrative.

Nella fase attuale della nostra ricerca si punta solo ad individuare opportuni strumenti tecnici, rimandando al dopo la necessaria progettazione di moduli didattici e la relativa sperimentazione.

2 Un esempio

Per comprendere meglio quanto si vuole fare, partiamo da un esempio di "teoria" nel linguaggio naturale. Supponiamo di essere a conoscenza dei seguenti "fatti":
Carlo ama Giovanni. Maria è una donna. Maria è allegra. Luisa è una donna. Luisa è giovane. Luisa è allegra.
Supponiamo, inoltre, di conoscere le seguenti "regole":
Ognuno ama se stesso. Se una persona è donna ed è simpatica, allora Carlo la ama. Se una persona è Italiana, allora è simpatica. Se una persona è allegra e giovane, allora è simpatica.
E' possibile riscrivere il tutto con il formalismo della logica matematica ottenendo la seguente teoria del primo ordine:

***P*:**

ama(Carlo, Giovanni)
ama(Carlo, *y*)←*donna*(*y*) ∧ *simpatica*(*y*)
ama(*x*, *x*)
donna(Maria)
donna(Luisa)
giovane(Luisa)
allegra(Luisa)
allegra(Maria)
simpatica(*x*) ←*Italiana*(*x*)
simpatica(*x*) ←*allegra*(*x*) ∧ *giovane*(*x*).

Come al solito, nelle formule con variabili libere è sottointesa la quantificazione universale. Indicheremo con *P* una tale teoria. A questo punto, è possibile chiedersi:

- data un'asserzione *A*, è possibile provare che *A* è conseguenza logica della teoria?

Ad esempio, ci si può chiedere se "*ama*(Carlo, Luisa)" o se "*ama*(Carlo, Maria)" siano teoremi. Naturalmente, utilizzando la logica a livello intuitivo, è facile vedere che nel primo caso la risposta è positiva, nel secondo caso la risposta è negativa. Si può, infatti, argomentare nel modo seguente. Dalla prima regola segue che Carlo *ama y* se *y* è *donna* ed è *simpatica*. Continuando a scorrere la nostra teoria, leggiamo che "*donna*(Luisa)" è un fatto di cui siamo già a conoscenza ed, inoltre, sappiamo anche che sono veri "*giovane*(Luisa)" ed "*allegra*(Luisa)". L'ultima regola della teoria afferma che se una *x* risulta essere "*allegra*" e "*giovane*", allora tale *x* sarà "*simpatica*". Dunque, Luisa risulta essere "*donna*" e "*simpatica*" e ciò fa scattare la prima regola; quindi, possiamo dedurre *ama*(Carlo, Luisa). Un pò più difficile risulta mostrare che *ama*(Carlo, Maria) non è un teorema. Da un punto di vista teorico si dovrebbe mostrare che esiste almeno un modello del sistema di assiomi che non è un modello dell'asserzione *ama*(Carlo, Maria). Questo è quanto viene fatto ad esempio quando si

dimostra che l'assioma delle parallele è indipendente dal rimanente sistema di assiomi della geometria euclidea. Qui ci limitiamo ad osservare che non ci sono elementi sufficienti a far scattare la prima regola e, dunque, non sarà possibile dedurre tale fatto dalla teoria.

Più in generale, stante la semplicità della teoria non è difficile verificare che sono teoremi di *P* tutti e soli i seguenti fatti:
ama(Carlo, Giovanni); *donna*(Maria); *donna*(Luisa); *giovane*(Luisa); *allegra*(Luisa); *allegra*(Maria); *ama*(Carlo, Carlo); *ama*(Giovanni, Giovanni); *ama*(Maria, Maria); *ama*(Luisa, Luisa); *simpatica*(Luisa); *ama*(Carlo, Luisa).

Questo procedimento potrebbe rimanere anche a livello intuitivo e, d'altra parte, dal punto di vista didattico, il ragionamento informale è insostituibile. Tuttavia, noi riteniamo che sia utile individuare metodi formali ed ambienti informatici che siano di base teorica e di supporto per affrontare un tale tipo di attività didattica. Si tratta, allora, di trasformare tale procedere intuitivo in un vero e proprio processo di calcolo e di verifica. Ci sembra, infatti, che la questione del rapporto tra intuizione e formalizzazione, che è uno dei punti fondamentali della ricerca didattica in ogni settore della matematica, si debba porre anche per la nozione di dimostrazione.

Naturalmente, com'è noto, esistono già uno strumento teorico ed uno informatico che appaiono adeguati allo scopo. Si tratta della programmazione logica e del linguaggio *Prolog* ad essa legato. Tuttavia, noi vogliamo individuare tecniche che permettano di costruire "dimostratori di teoremi" in ambienti, quali ad esempio *Mathematica*, meno specifici del *Prolog*. Questo ultimo linguaggio, infatti, non ha la duttilità degli altri linguaggi ed è, attualmente, per lo più diffuso solo nelle università.

3 Trasformare una teoria

Riferendoci alla tradizione della programmazione logica, considereremo solo teorie costituite da "fatti" e "regole". Ricordiamo che si chiama *fatto* una formula atomica chiusa. Si chiama *regola* una formula atomica oppure una formula del tipo $\alpha_1 \wedge ... \wedge \alpha_m \rightarrow \alpha$ con α, $\alpha_1,...,\alpha_m$ formule atomiche non necessariamente chiuse. Un *programma definito* è una teoria costituita solo da fatti e regole. Spesso una regola viene scritta nel modo seguente $\alpha \leftarrow \alpha_1 \wedge ... \wedge \alpha_m$; la formula α è detta *testa* e la formula $\alpha_1 \wedge ... \wedge \alpha_m$ *corpo* della regola. Un'ulteriore restrizione che faremo consiste nel considerare solo regole in cui tutte le variabili che compaiono nel corpo compaiono anche nella testa. Nel seguito giustificheremo tale restrizione e vedremo come si possa trattare il caso più generale. Infine, per quanto riguarda il tipo di asserzioni di cui si vuole trovare la dimostrazione, ci limiteremo solo ai "fatti". Una volta fatte tali scelte, interpretiamo ogni predicato come una funzione a valori nell'algebra di Boole {*Falso*, *Vero*} e ci proponiamo di trasformare il sistema di assiomi in un insieme di definizioni di funzioni. Un'interrogazione consisterà, a questo punto, nel semplice calcolo del valore di una funzione.

Vediamo, passo per passo, come si può attuare tale trasformazione. La prima cosa da fare è:

- individuare regole di riscrittura, che trasformino il programma di partenza in un programma equivalente, le cui regole abbiano una testa in cui compaiono solo variabili distinte e nessuna costante od operazione.

La cosa può essere realizzata nel modo seguente:

R1. data una formula atomica $r(t_1, \ldots, t_n)$, con $t_1,...,t_n$ termini, ad essa viene sostituita la formula logicamente equivalente

$$r(x_1, \ldots, x_n) \leftarrow (x_1 = t_1) \wedge ... \wedge (x_n = t_n)$$

R'1. data una formula del tipo $r(t_1, \ldots, t_n) \leftarrow corpo$, con $t_1,\ldots,t_n$ termini, ad essa viene sostituita la formula logicamente equivalente

$$r(x_1, \ldots, x_n) \leftarrow (x_1 = t_1) \wedge \ldots \wedge (x_n = t_n) \wedge corpo.$$

Nel caso della teoria *P* descritto nel paragrafo precedente questa prima trasformazione porta alla seguente equivalente teoria, che indichiamo con *P'*:

***P'*:**

$ama(x, y) \leftarrow y = x$
$ama(x, y) \leftarrow (x = \text{Carlo}) \wedge (y = \text{Giovanni})$
$ama(x, y) \leftarrow (x = \text{Carlo}) \wedge donna(y) \wedge bello(y)$
$donna(x) \leftarrow (x = \text{Maria})$
$donna(x) \leftarrow (x = \text{Luisa})$
$giovane(x) \leftarrow (x = \text{Luisa})$
$allegra(x) \leftarrow (x = \text{Luisa})$
$simpatica(x) \leftarrow Italiana(x)$
$simpatica(x) \leftarrow allegra(x) \wedge giovane(x)$

Possiamo ora attuare una successiva trasformazione, che fa riferimento all'osservazione per cui un insieme di formule del tipo

$$\{A \leftarrow corpo_1, \ldots, A \leftarrow corpo_k\}$$

è equivalente all'unica formula

$$A \leftarrow corpo_1 \vee \ldots \vee corpo_k.$$

In base a tale osservazione, sia *r* un predicato tale che in *P'* ci siano esattamente *k* regole la cui testa sia $r(x_1, \ldots, x_n)$:

$r(x_1, \ldots, x_n) \leftarrow corpo_1$
$r(x_1, \ldots, x_n) \leftarrow corpo_2$
$\vdots$
$r(x_1, \ldots, x_n) \leftarrow corpo_k$

Possiamo allora sostituire tale insieme di regole con la formula

$$r(x_1, \ldots, x_n) \leftarrow corpo_1 \vee \ldots \vee corpo_k.$$

Applicando questa trasformazione ad ogni predicato che compare come testa di una regola, la nostra teoria *P'* diventa

***P''*:**

ama(*x*, *y*) ← ((*y*=*x*) ∨ ((*x*=Carlo)∧(*y*=Giovanni)) ∨
∨ (((*x*=Carlo)∧*donna*(*y*)∧*simpatica*(*y*)))
donna(*x*) ← ((*x* = Maria) ∨ (*x* = Luisa))
giovane(*x*) ← (*x* = Luisa)
alto(*x*) ← (*x* = Luisa)
simpatica(*x*) ← (*Italiano*(*x*) ∨ (*allegra*(*x*) ∧ *giovane*(*x*)))

Naturalmente tale teoria, che abbiamo indicato con *P"*, è ancora logicamente equivalente a *P*.

4 Dalla logica alla programmazione funzionale

Il passo finale del processo dovrebbe essere la "traduzione", in un linguaggio di programmazione, di ogni definizione di un predicato in una definizione di funzione a valori in {*Falso*, *Vero*}. Vediamo, ad esempio, come si può "tradurre" la nostra teoria nel caso del linguaggio *Mathematica*, in modo da ottenere una "deduzione automatica".

Riportiamo il codice, scritto secondo la sintassi di *Mathematica*, in cui:
i) i simboli || e && traducono, rispettivamente, la disgiunzione e la congiunzione,
ii) il simbolo = = = rappresenta la relazione d'identità,
iii) il simbolo := rappresenta l'istruzione di assegnazione.

In *Mathematica*:

ama[*x*_,*y*_] := (*y* = = = x) || (*x* = = = carlo && *y* = = = giovanni)

```
                || (x = = = maria && y = = = maria) || ((x =
                = = = carlo) && donna[y] && bello[y]);
donna[x_] := (x = = = maria) || (x = = = luisa);
giovane[x_] := (x = = = luisa);
alto[x_] := (x = = = luisa);
Italiano[x_] := False
```

E' evidente che un tale tipo di programma può essere scritto in un qualunque linguaggio di programmazione in quanto in ogni linguaggio è possibile definire le funzioni ed in ogni linguaggio è possibile trattare i valori Booleani. D'altra parte, uno degli scopi della nostra ricerca è di svincolare l'implementazione di un "dimostratore" di teoremi da un particolare linguaggio.

Se ora vogliamo conoscere il valore di verità della formula "*ama*(Carlo, Giovanni)" basta calcolare il valore della funzione *ama* in corrispondenza dell'input (Carlo, Giovanni). In pochi passi *Mathematica* restituisce il valore *True*. Quando, invece, chiediamo "*ama*(Carlo, Maria)", *Mathematica* risponderà con *False*. Tuttavia, come vedremo nel prossimo paragrafo, è necessario interpretare opportunamente il significato della risposta *False*. Da un punto di vista logico, in realtà, la risposta esatta dovrebbe essere del tipo "non è dimostrabile".

5 La traduzione è logicamente equivalente alla teoria?

E' facile vedere che il sistema di definizioni di funzioni in cui abbiamo tradotto la nostra teoria non è equivalente alla teoria stessa. Ad esempio, se si confronta la formula

$$donna(x) \leftarrow ((x = = = \text{Maria}) \vee (x = = = \text{Luisa}))$$

con la sua "traduzione"

donna[x_] := (x = = = Maria) || (x = = = Luisa)

ci si accorge che, mentre nel primo caso niente giustifica la conclusione che Maria e Luisa siano le uniche donne, la

traduzione invece comporta proprio tale conclusione. Infatti, ad una domanda del tipo "*donna*(Carla)?" *Mathematica* risponderebbe "False", in quanto sono false sia la proposizione (Carla = = = Maria) che la proposizione (Carla = = = Luisa). Invece, quando si assume l'implicazione, quello che avviene è che non si riesce a dedurre *donna*(Carla), ma nemmeno si dimostra esplicitamente che tale asserzione sia falsa.
In realtà, la traduzione fatta equivale all'equivalenza logica

$$donna(x) \leftrightarrow ((x = = = \text{Maria}) \vee (x = = = \text{Luisa}))$$

Più in generale, è possibile dimostrare che l'insieme di definizioni di funzioni in cui abbiamo tradotto la teoria *P* non è equivalente a *P*, ma ad un diverso sistema di assiomi che prende il nome di *completamento di Clark* di *P* la cui definizione rigorosa è la seguente:

Definizione. Dato il programma *P*, il *completamento di Clark* è il programma *Clark*(*P*) che si ottiene sostituendo in *P''* ogni implicazione del tipo

$$r(x_1, \ldots, x_n) \leftarrow corpo_1 \vee \ldots \vee corpo_k$$

con l'equivalenza

$$r(x_1, \ldots, x_n) \leftrightarrow corpo_1 \vee \ldots \vee corpo_k,$$

ed aggiungendo la formula

$$\neg r(x_1, \ldots, x_n)$$

per ogni predicato *r* presente in *P* che non sia la testa di una regola in *P''*.

Per ovvi motivi, la formula $r(x_1, \ldots, x_n) \leftrightarrow corpo_1 \vee \ldots \vee corpo_k$, oppure la formula $\neg r(x_1, \ldots, x_n)$ vengono chiamate *definizione del predicato r*. Nel nostro caso, la traduzione effettuata del programma *P* corrisponde al sistema di assiomi seguente:

$$ama(x, y) \leftrightarrow ((y = x) \vee ((x = \text{Carlo}) \wedge (y = \text{Giovanni})) \vee ((x = \text{Maria}) \wedge (y = \text{Maria})) \vee ((x = \text{Carlo}) \wedge donna(y) \wedge bello(y)))$$

$$donna(x) \leftrightarrow ((x = \text{Maria}) \vee (x = \text{Luisa}))$$

giovane(*x*) ↔ (*x* = Luisa)
alto(*x*) ↔ (*x* = Luisa)
¬*Italiano*(*x*)

6 *"Non dimostrabile"* al posto di *"Falso"*

Poniamo ancora l'accento sul fatto che la teoria *Clark*(*P*), ottenuta nel paragrafo precedente, non è più equivalente a *P*. La differenza più importante è che, in *Clark*(*P*), se il corpo è falso, anche la testa lo è. Ciò equivale ad affermare che se non abbiamo abbastanza elementi per dimostrare un fatto, allora diciamo che esso è falso. Questo modo di procedere è accettato comunemente nella programmazione logica ed è collegato ad uno dei suoi problemi più controversi, che è l'accettazione del connettivo "negazione" nel corpo delle regole. In tale caso per dimostrare la negata di una formula si usa la regola della "negazione come fallimento" che consiste nell'affermare che un fatto *X* è falso se esso non è conseguenza logica del programma, vale a dire, se ogni possibile dimostrazione di *X* fallisce.

Anche se la negazione come fallimento è una regola comunemente utilizzata in programmazione logica, da un punto di vista didattico potrebbe creare notevoli confusioni. Pensiamo, infatti, che sia essenziale evitare di confondere il "non dimostrabile" con il "falso". Tenendo conto di ciò, si potrebbe apportare una modifica al programma, che permetta di "trasformare il *falso* in *non dimostrabile*". Ad esempio, dopo la definizione della funzione "ama", che rimane la seguente

```
ama[x_,y_] := (y = = = x) || (x = = = carlo && y = = =
                giovanni) || (x = = = maria && y = = =
                maria) || ((x = = = carlo) && donna[y] &&
                bello[y]);
```

si potrebbe definire la seguente funzione “Teorema”:

Teorema[x_]:=If[x= = =True, "è un teorema", "non è dimostrabile, a partire da ciò di cui siamo a conoscenza"]

A questo punto le domande si pongono nel modo seguente:

Teorema[ama[carlo,luisa]]

Teorema[ama[giovanni,luisa]]

e, come risposte, avremmo "è un teorema" nel primo caso, "non è dimostrabile, a partire da ciò di cui siamo a conoscenza", nel secondo caso.

7 Il problema del quantificatore esistenziale

Esaminiamo ora cosa succede se nel nostro programma compare una regola del tipo

$$simpatizza(x, y) \leftarrow ama(x, z) \wedge ama(z, y),$$

cioè una regola in cui ci sono variabili (z in questo caso) che compaiono nel corpo, ma non nella testa. Se si provasse a tradurre tale tipo di regola come fatto fino ad ora, si otterrebbe, in *Mathematica*

simpatizza[x_,y_] := ama[x, z] && ama[z, y].

Tuttavia, se interroghiamo il programma ottenuto in questo modo, non otteniamo da esso le risposte che ci aspettiamo. Ad esempio, se chiedessimo

simpatizza[carlo, luisa]

otterremmo, come risposta, “False”, mentre dovrebbe essere “True”. Il motivo è che z è una variabile che non viene inizializzata.

In questo caso, per raggiungere il nostro scopo, ossia ottenere un insieme di definizioni di funzioni, è necessaria

un'ulteriore trasformazione. Tale trasformazione fa riferimento al fatto che la formula

$$A(x) \leftarrow A(z),$$

dove le variabili si sottintendono quantificate universalmente, ovvero $\forall(x)\forall(z)(A(x) \leftarrow A(z))$, è equivalente alla formula

$$A(x) \leftarrow \exists(z)A(z).$$

Allora, la nostra regola di riscrittura è la seguente

R2. Data una clausola $r(t_1, \ldots, t_n) \leftarrow$ *corpo*, se y è una variabile presente nel *corpo*, ma non in $r(t_1, \ldots, t_n)$, allora dobbiamo riscrivere $r(t_1, \ldots, t_n) \leftarrow$ *corpo* nella formula logicamente equivalente

$$r(t_1, \ldots, t_n) \leftarrow \exists y(corpo).$$

Con questa trasformazione si ottiene una teoria equivalente a quella di partenza. Il problema consiste ora nel valutare una formula come $\exists y(corpo)$ in cui è presente un quantificatore esistenziale. In *Mathematica* ciò si può ottenere definendo prima il dominio U ed il suo prodotto cartesiano:

```
U:={carlo,giovanni,luisa,maria};
Cartesian[l_,m_]:=Flatten[Outer[List,l,m],1];
UxU:=Cartesian[U,U];
```

Si passa poi a definire una estensione della disgiunzione:

```
Vel[{X_,Y_}]:=Or[X,Y];Vel[L_]:=Or[First[L],Vel[Rest[L]
]];
```

Infine, si ottiene il quantificatore esistenziale:

```
Esiste[x_,Universo_,A_]:=Vel[Map[Function[x,A],
                                 Universo]];
```

Possiamo, pertanto, osservare che la nostra regola alla fine può essere tradotta nella seguente definizione di funzione:

simpatizza[x_,y_]:=Esiste[z,U,Unevaluated[ama[x,z]&& ama[z,y]]]

A questo punto, alla domanda

simpatizza[carlo, luisa]

il programma risponde correttamente, ovvero "True".

Bibliografia

[1] Clark K. L., Negation as Failure, in *Logic and Data Bases*, Gallaire H. and Minker J. (eds), Plenum Press, New York, 1978, pp.293-322.

[2] Fitting M., Fixpoint semantics for logic programming a survey, *Theoretical Computer Science*, 278, (2002), pp. 25-51.

[3] Gerla G., Fuzzy Logic Programming and Fuzzy Control, *Studia Logica*, 79, (2005), pp. 231-254.

[4] Lloyd J. W., *Foundations of Logic Programming*, Springer-Verlag, 1987